Modeling Randomized Data Streams in Caching, Data Processing, and Crawling Applications

Sarker Tanzir Ahmed and Dmitri Loguinov

Internet Research Lab
Department of Computer Science and Engineering
Texas A&M University

April 29, 2015

Agenda

- Introduction
- Analysis of 1D Streams
  - LRU Performance
  - MapReduce Disk I/O
- Analysis of 2D Streams
  - Properties of the Seen Set, Discovered Nodes, and the Frontier
- Conclusion

Introduction

- Key-value input pairs are common to MapReduce and many other types of applications
- Input is typically a finite length stream where the keys come off a finite set
- Experience of the processing application (e.g., RAM/disk usage, processing speed) depends largely on the properties of the stream (i.e. key frequency)
- Example: Least Recently Used (LRU) cache's hit rate is governed by popularities of items

Introduction (2)

- MapReduce applications' combined output (the result of merging duplicate keys in a window of pairs)
  - Depends on the frequency properties of the keys
  - Usually, the higher the frequencies of each item, the smaller the size of the combined output
- Existing literature is missing accurate model
  - Common to assume linear ratio between input and output

Agenda

- Introduction
- Analysis of 1D Streams
  - LRU Performance
  - MapReduce Disk I/O
- Analysis of 2D Streams
  - Properties of the Seen Set, Discovered Nodes, and the Frontier
- Conclusion

Analysis of 1D Streams

- Define one-dimensional (1D) streams as discrete-time processes $\{Y_t\}_{t\geq 1}$, where each item Y_t is observed at t
  - Y_t is unique (i.e., previously unseen) with probability $p(t)$ (also called *uniqueness probability)*, and duplicate otherwise
- Input is a stream of length T
  - Keys belong to a finite set V of size n
  - Each key v is repeated $\mathcal{I}(v)$ times (random variable $\mathcal{I}$ also denotes frequency distribution of v's)
  - The seen set at t is denoted by S_t, and the unseen set by U_t
- We also assume uniform shuffle of the items across the stream
  - Independent Reference Model (IRM)

Analysis of 1D Streams (2)

- Theorem 1: The probability of seeing a unique (previously unseen) key at t (using $\epsilon_t = t/T$) is:

$$p(t) = \frac{1}{E[\mathcal{I}]} E\left[\mathcal{I} \cdot (1-\epsilon_t)^{\mathcal{I}-1}\right]$$

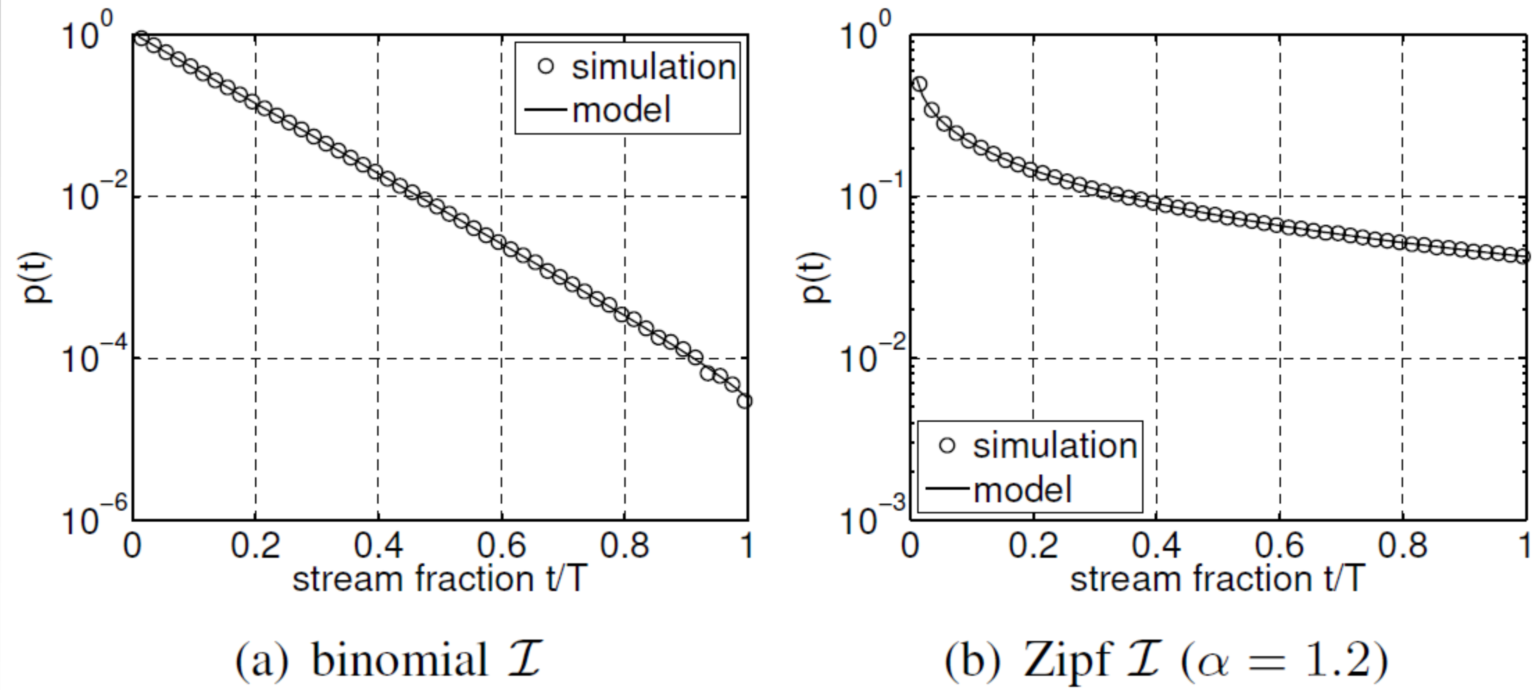


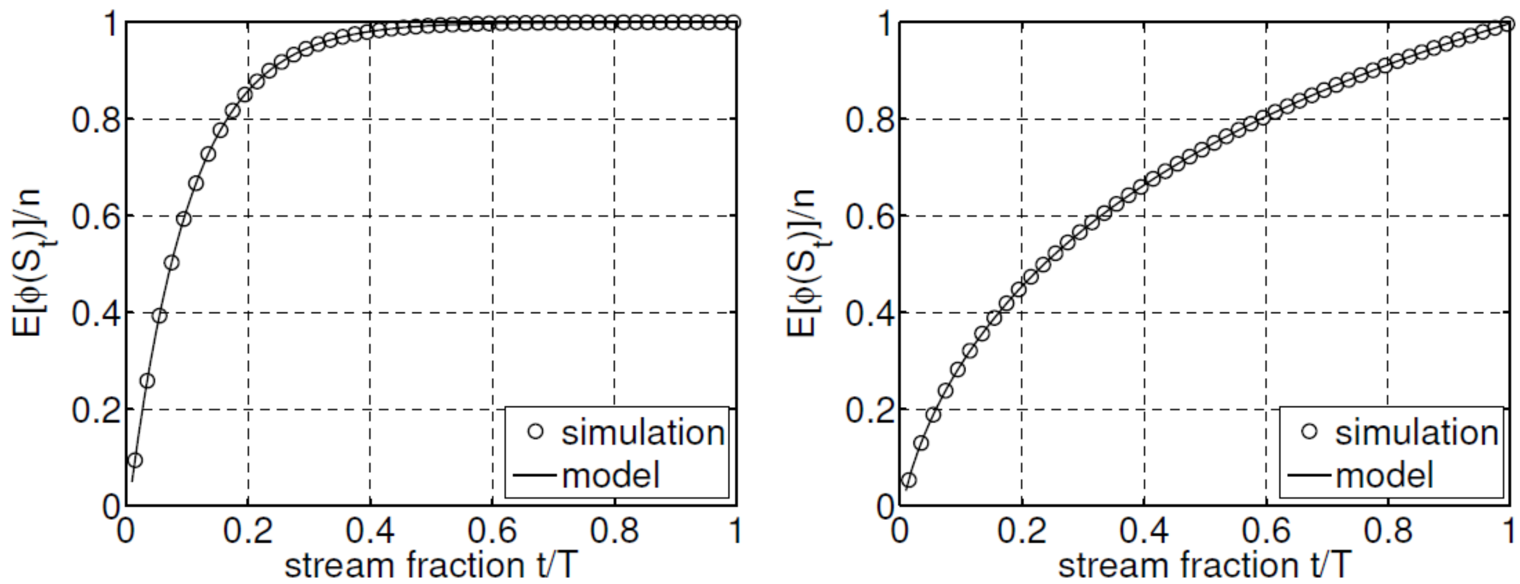
Fig: Verification of $p(t)$ under $E[\mathcal{I}]=10$, $n=10\mathrm{K}$.

Computer Science, Texas A&M University

Analysis of 1D Streams (3)

- Theorem 2: The size of the seen set at t after using $|A|=\phi(A)$ is:

$$E[\phi(S_t)] = nE[1-(1-\epsilon_t)^{\mathcal{I}}]$$



(a) binomial $\mathcal{I}$ (b) Zipf $\mathcal{I}$ ($\alpha = 1.2$)

Fig: Verification of the size of S_t ($E[\mathcal{I}]=10$, $n=10\text{K}$).

1D Streams - Applications

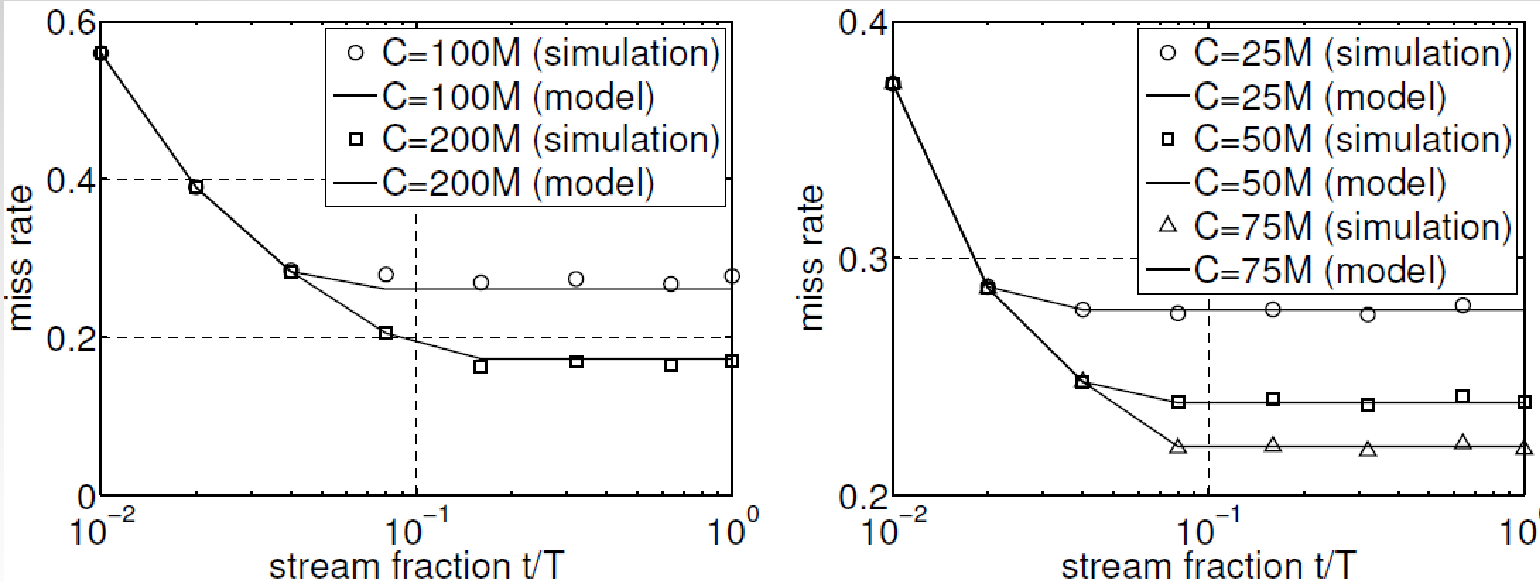
- We consider two applications:
  - Miss rate of LRU cache
  - Disk I/O of MapReduce
- For verification, we use the following two workloads in addition to simulated input:
  - IRLbot host graph (640M nodes, 6.8B edges, 55 GB)
  - WebBase web graph (635M nodes, 4.2B edges, 35 GB)

LRU Cache Miss Rate

- Theorem 3: The miss rate of a LRU cache of size C is:

$$m(t) = \frac{1}{E[\mathcal{I}]} E[\mathcal{I}(1-\epsilon_{\min(t,\tau)})^{\mathcal{I}-1}].$$

Here, the value τ is obtained as $f^{-1}(C)$, where $f(t) = E[\phi(S_t)]$



(a) IRLbot host graph (b) WebBase web graph

Fig. Verification of LRU miss rate in real graphs.

1D Streams - MapReduce Disk I/O

- Input is a stream of length T
  - Entries are key-value pairs, each $K+D$ bytes
  - At time step t, one pair is processed by MapReduce
- Disk I/O consists of:
  - Input with T pairs (some duplicate)
  - Output with n unique pairs
  - Sorted runs of size L
- Total disk overhead is $W = (K+D)(T + n + 2L)$
  - Our goal is to derive L
- RAM can hold m pairs in a merge-sort MapReduce
  - Then, $k = \lceil T/m \rceil$ is the number of sorted runs, where each contains $E[|S_m|]$ pairs on average

MapReduce Disk I/O (2)

- Theorem 4: Disk spill L of a merge-sort MapReduce is:

$$L = nk(K+D)\left(1 - E\left[(1-\epsilon_m)^{\mathcal{I}}\right]\right),$$

And, the total disk I/O is thus:

$$W = n(K+D)\left\{E[\mathcal{I}] + 1 + 2k\left(1 - E\left[(1-\epsilon_m)^{\mathcal{I}}\right]\right)\right\}.$$

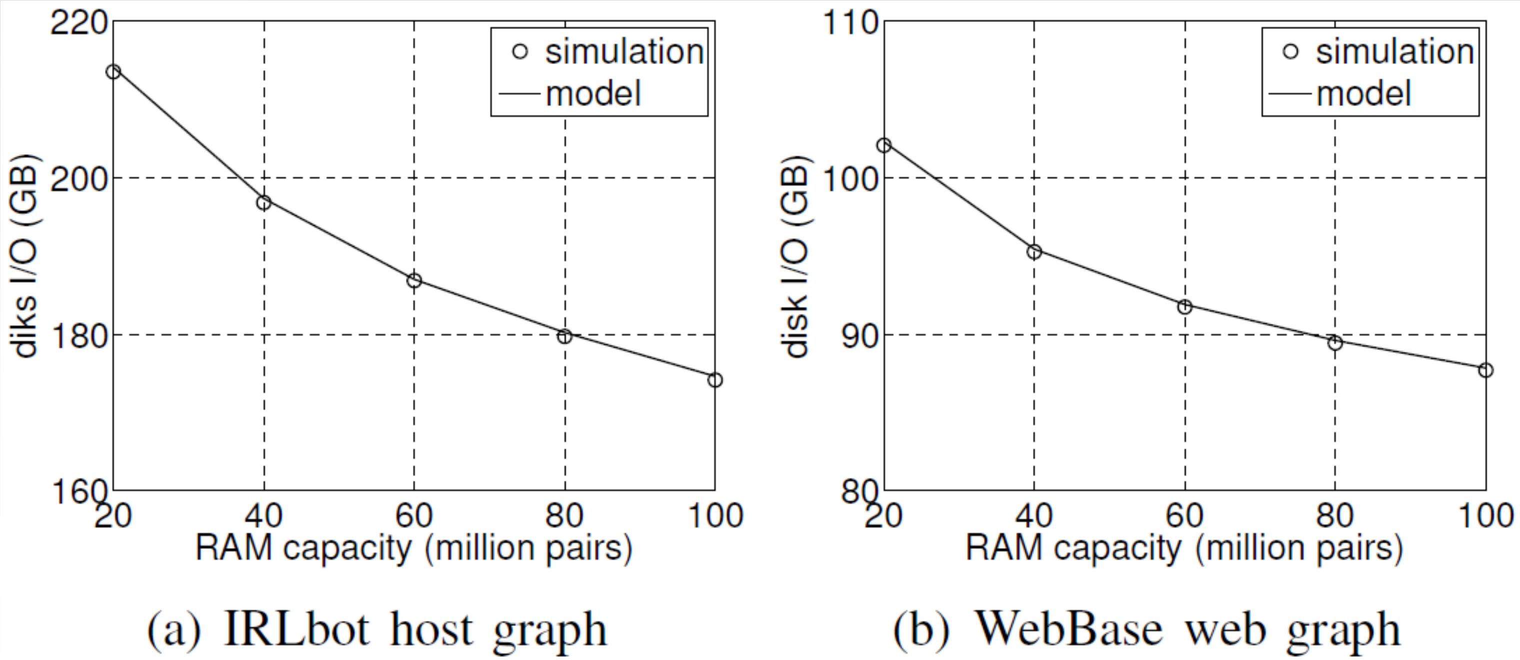


Fig: Verification of Disk I/O of a merge-sort MapReduce.

Agenda

- Introduction
- Analysis of 1D Streams
  - LRU Performance
  - MapReduce Disk I/O
- Analysis of 2D Streams
  - Properties of the Seen Set, Discovered Nodes, and the Frontier
- Conclusion

Analysis of 2D Streams

- Two-dimensional (2D) streams are mainly applicable in analyzing graph traversal algorithms
- Consider a simple directed random graph $G(V,E)$
  - V and E are the set of nodes and edges, respectively. Let $|E| = T$ and the in/out-deg sequences be $\{\mathcal{I}(v)\}_{v\epsilon V}$ and $\{\mathcal{O}(v)\}_{v\epsilon V}$
- Define the stream of edges of this graph seen by a crawler as a 2D discrete-time process $\{(X_t, Y_t)\}^T{}_{t=1}$
  - Here X_t is the crawled node and Y_t the destination node
- Define the crawled set as $C_t = \cup_{i=1}^t\{X_i\}$, the seen set as $S_t = \cup_{i=1}^t\{Y_i\}$, and the frontier as $F_t = S_t \setminus C_t$
- The goal is to analyze the stream of Y_t's, and the sets S_t and F_t as they change over crawl time t

Analysis of 2D Streams (2)

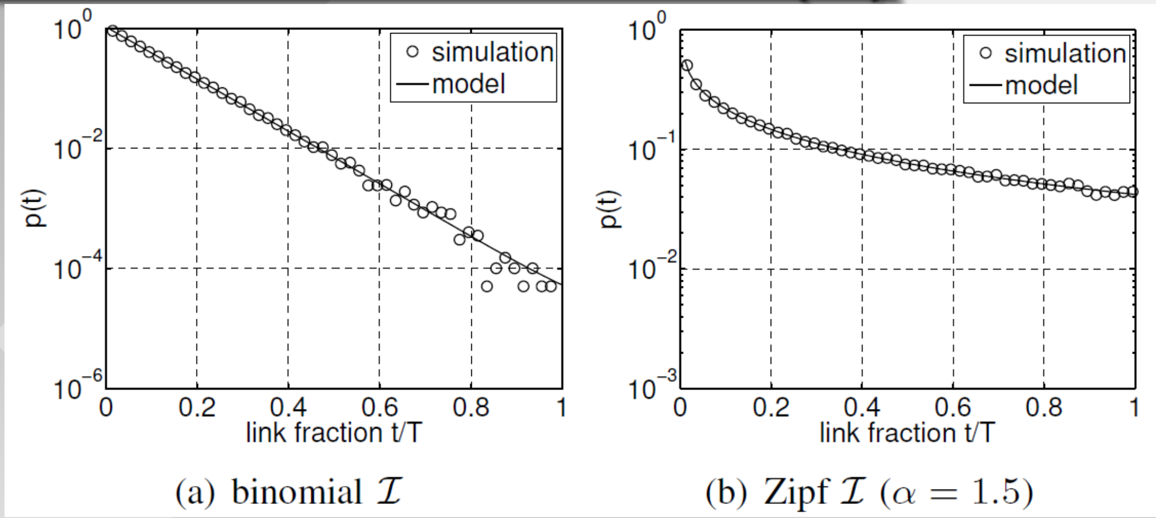


Fig. Verification of $p(t)$ under BFS crawl on graph ($E[\mathcal{I}]$=10, n=10K).

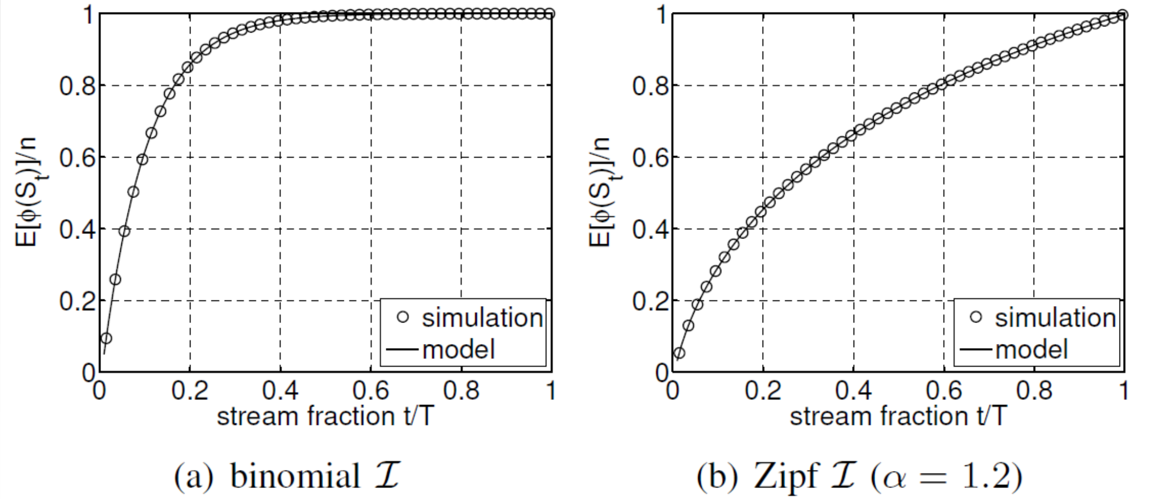


Fig. Verification of the seen set size in BFS ($E[\mathcal{I}]$=10, n=10K).

Seen Set Properties

- Theorem 5: The average in/out-degree of the nodes in the seen set:

$$\bar{\mathcal{I}}(S_t) \approx \frac{E[\mathcal{I} \cdot (1-(1-\epsilon_t)^{\mathcal{I}})]}{1-E[(1-\epsilon_t)^{\mathcal{I}}]},$$

$$\bar{\mathcal{O}}(S_t) \approx \frac{E[\mathcal{O} \cdot (1-(1-\epsilon_t)^{\mathcal{I}})]}{1-E[(1-\epsilon_t)^{\mathcal{I}}]}.$$

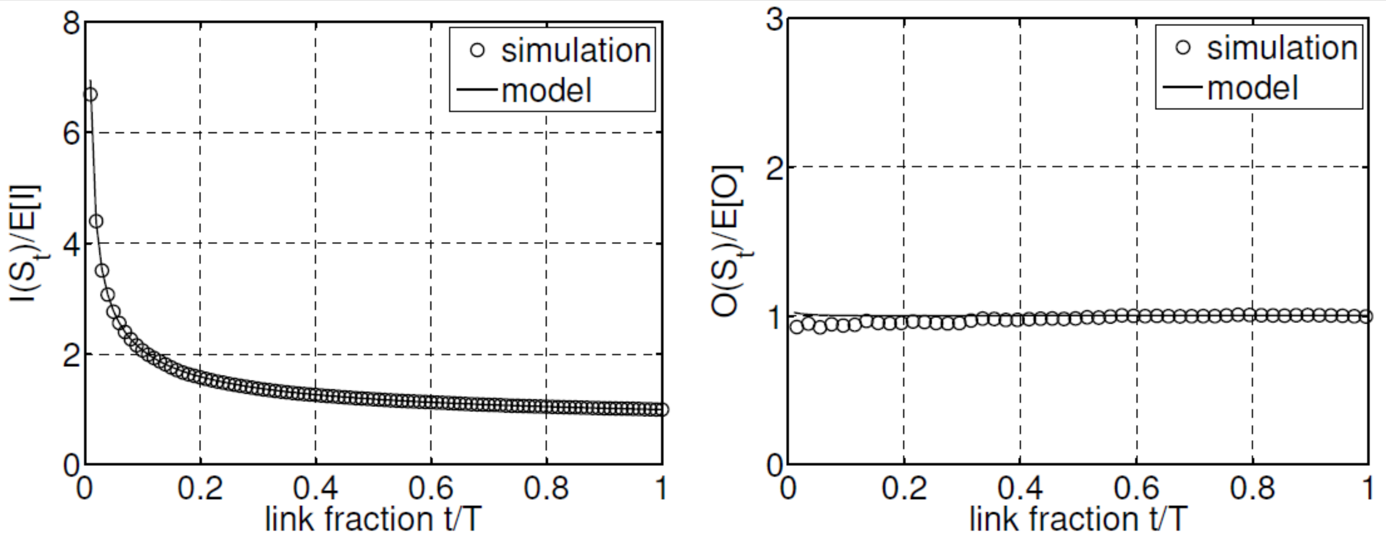


Fig: Verification of the average in/out-degree of the seen set

Destination Node Properties

- Theorem 6: The in-degree distribution of the Y_t is :

$$P(\mathcal{I}(Y_t) = k) = \frac{kP(\mathcal{I} = k)}{E[\mathcal{I}]}.$$

  - Helps obtain an unbiased estimator of $P(\mathcal{I}{=}k)$ after observing m edges: $\frac{\sum_{t=1}^{m} \mathbf{1}_{\mathcal{I}(Y_t)=k}}{k\sum_{t=1}^{m} 1/\mathcal{I}(Y_t)}.$

- Theorem 7: The average in/out-degree of Y_t is independent of time and equals:

$$E[\mathcal{I}(Y_t)] = \frac{E[\mathcal{I}^2]}{E[\mathcal{I}]}, \qquad E[\mathcal{O}(Y_t)] = \frac{E[\mathcal{I}\mathcal{O}]}{E[\mathcal{I}]},$$

while that of Y_t, conditioned on its being unseen is:

$$E[\mathcal{I}(Y_t)|Y_t \in U_{t-1}] = \frac{E[\mathcal{I}^2 \cdot (1-\epsilon_t)^{\mathcal{I}-1}]}{E[\mathcal{I}(1-\epsilon_t)^{\mathcal{I}-1}]},$$

$$E[\mathcal{O}(Y_t)|Y_t \in U_{t-1}] = \frac{E[\mathcal{I}\mathcal{O} \cdot (1-\epsilon_t)^{\mathcal{I}-1}]}{E[\mathcal{I}(1-\epsilon_t)^{\mathcal{I}-1}]}.$$

Destination Node Properties - Verification

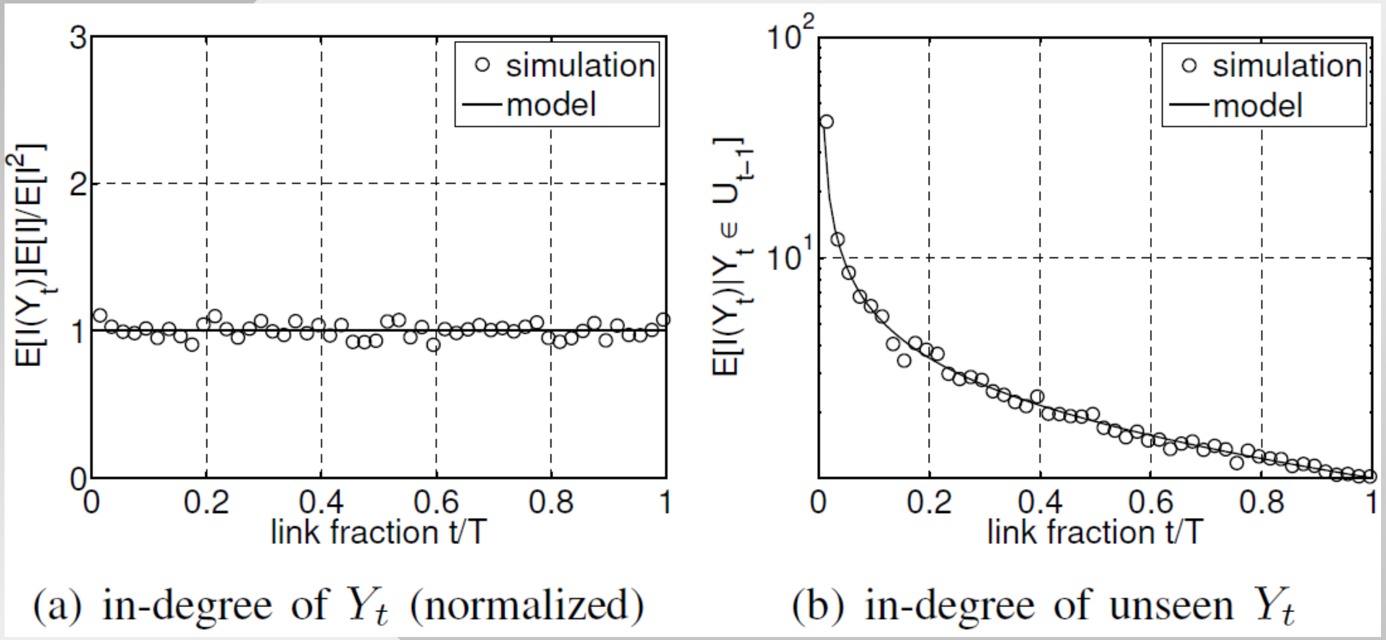


Fig: Verification of the average in-degree of all Y_t's and the unseen Y_t's, respectively

Properties of the Frontier

- A crawling method's efficiency is a function of the frontier size
  - The more the size, the more the load on duplicate-elimination, prioritization algorithms
- Theorem 8: The following iterative relation computes the size of the frontier (let $\phi(A)$=|A|):

$$E[\phi(F_t)] \approx E[\phi(F_{t-1})] + p(t-1) - \frac{1}{E[\mathcal{O}(X_{t-1})]}$$

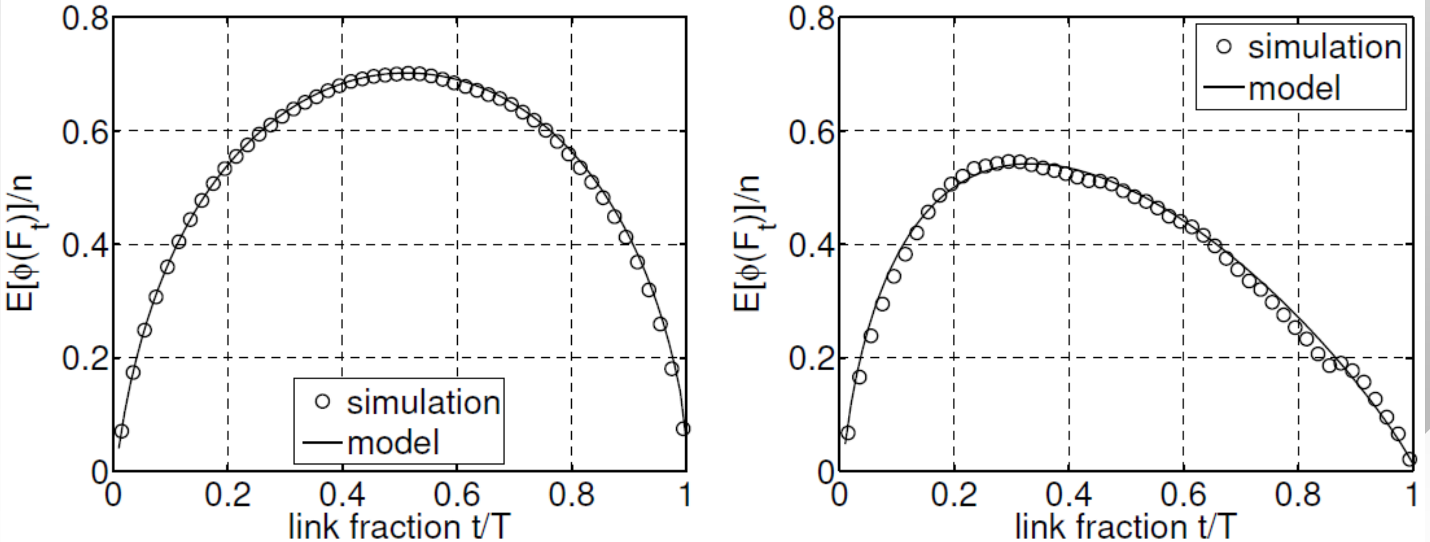
- We consider two crawling methods to examine their frontier sizes:
  - **B**reads **F**irst **S**earch (BFS)
  - **F**rontier **R**a**N**domization (FRN), where any node from the frontier is picked randomly for crawling

Properties of the Frontier (2)

- Theorem 9: For BFS, the out-degree of the crawled node is given by:

$$E[\mathcal{O}(X_{t+E[\mathcal{O}(F_t)]})] = \frac{E[\mathcal{I}\mathcal{O}(1-\epsilon_t)^{\mathcal{I}-1}]}{E[\mathcal{I}(1-\epsilon_t)^{\mathcal{I}-1}]},$$

while that for FRN is simply: $E[\phi(F_t)] = E[\mathcal{O}(F_t)]/E[\mathcal{O}]$.



(a) BFS (b) FRN

Fig: Verification of the frontier size with $E[\mathcal{I}]=10$ and $n = 10\text{K}$.

Agenda

- Introduction
- Analysis of 1D Streams
  - LRU Performance
  - MapReduce Disk I/O
- Analysis of 2D Streams
  - Properties of the Seen Set, Discovered Nodes, and the Frontier
- Conclusion

Conclusion

- Presented accurate analytic models of performance based on workload characterization
- Proposed a common modeling framework for a number of apparently unrelated fields (i.e., caching, MapReduce, crawl modeling)

Thank you!
Questions?